Shared Storage Resource Management Using Machine Learning and Control Theory

Mahmut Taylan Kandemir

(in collaboration with Padma Raghavan and Qian Wang)

The Pennsylvania State University

Department of Computer Science and Engineering

QoS Challenge in Storage

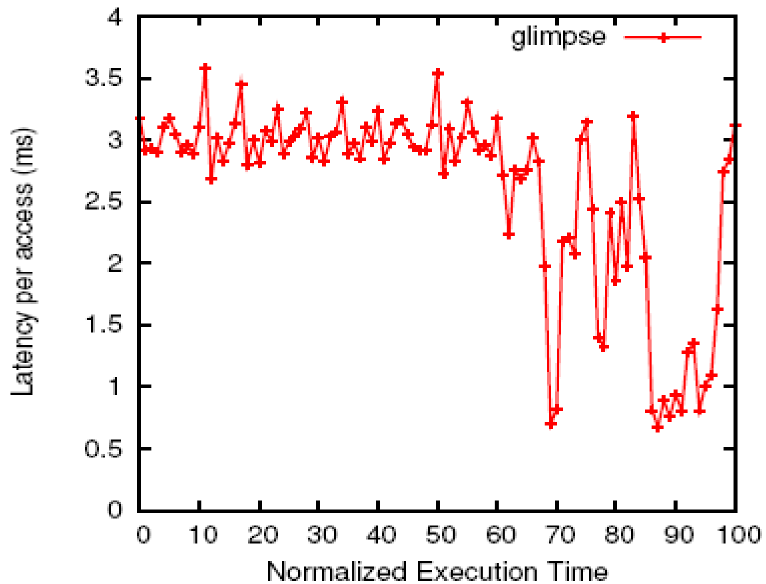
- Specification problem
  - How can we specify QoS?
    - Application level vs I/O level
    - Absolute vs relative
    - Hard vs soft
- Complex resource sharing patterns
  - Distributed resources, multi-server storage systems
  - Interactions among different types of resources
  - Hybrid storage architectures
- Measurement and enforcement
  - Where/how to enforce QoS?
  - Modeling and prediction [machine learning]
  - Tracking [formal feedback control]

Storage Cache Management Strategies

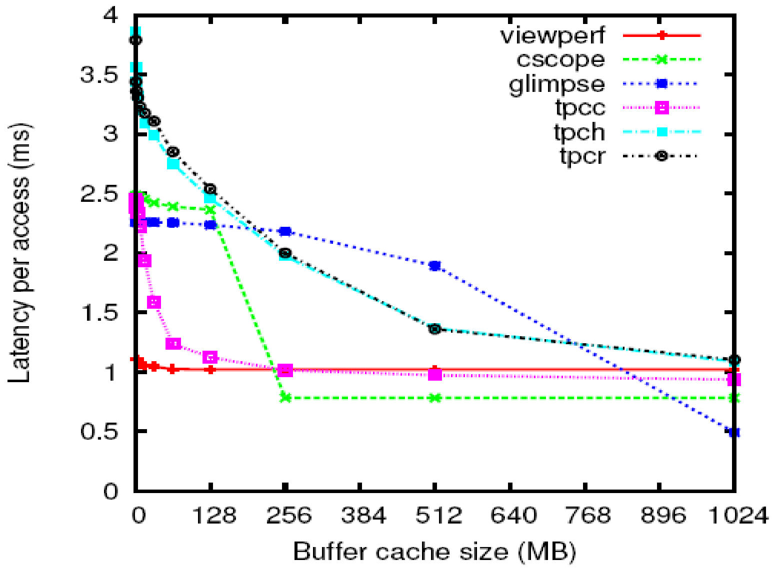
- No partitioning
  - Destructive interferences
- Equal (fair share) partitioning
  - Isolation, underutilization
- Static unequal partitioning
  - Isolation, lack of dynamic adaptation
- Dynamic adaptive partitioning

Motivation for Dynamic Scheme

Variation with Time



Variation with Cache Size



- Changing cache requirements of applications during execution
- Different saturating points with increasing cache sizes
- Motivates dynamically allocating cache based on application characteristics

Challenges in Dynamic Cache Space Management

- **Goals:** (1) Satisfy QoS for all applications
  (2) Improve performance

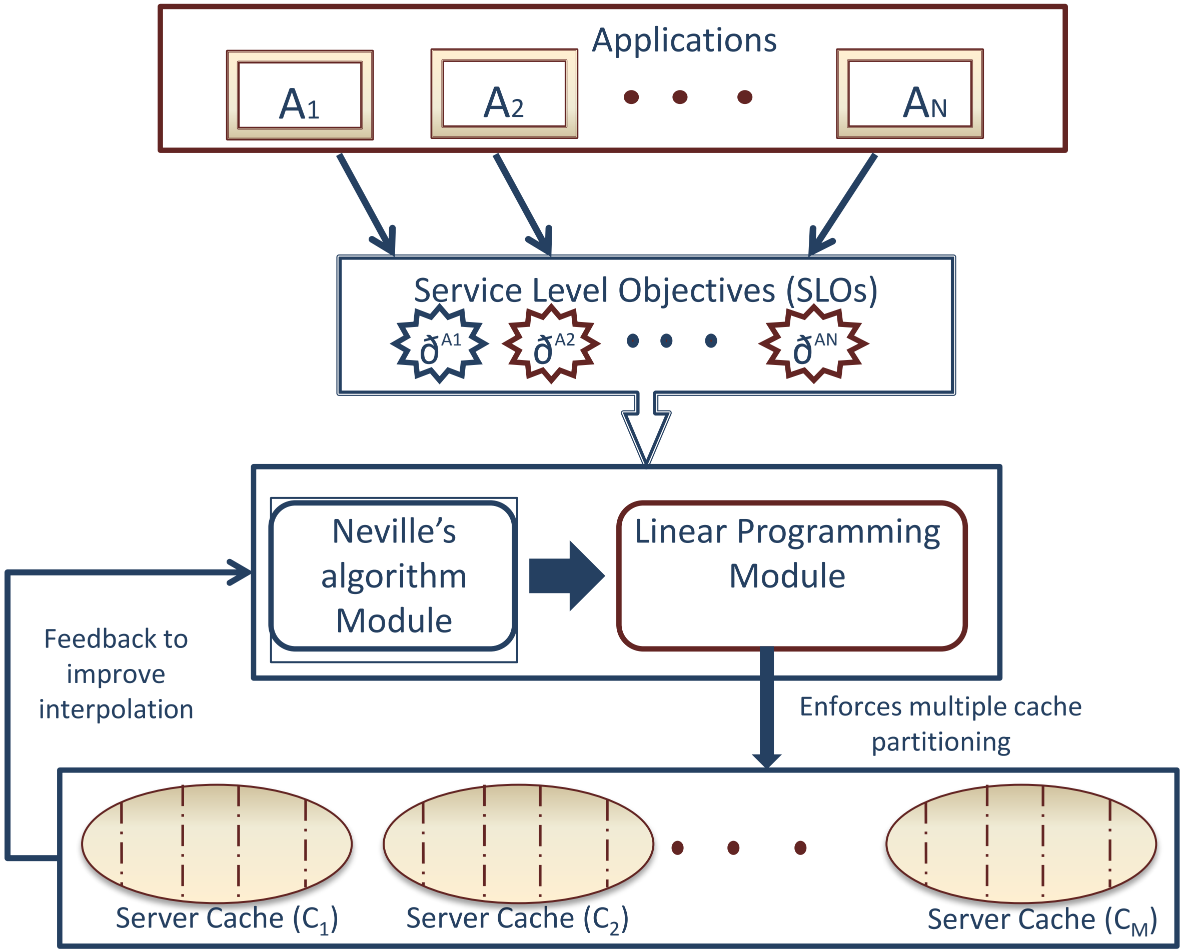
- How to partition the cumulative cache space across competing applications?
  (Neville's Algorithm)

- What would be the cache allocation across each of the available server caches for each application?
  (Linear Programming)

- How would the cache allocations adapt to dynamic modulations in cache requirements at runtime?
  (Feedback in every enforcement interval)

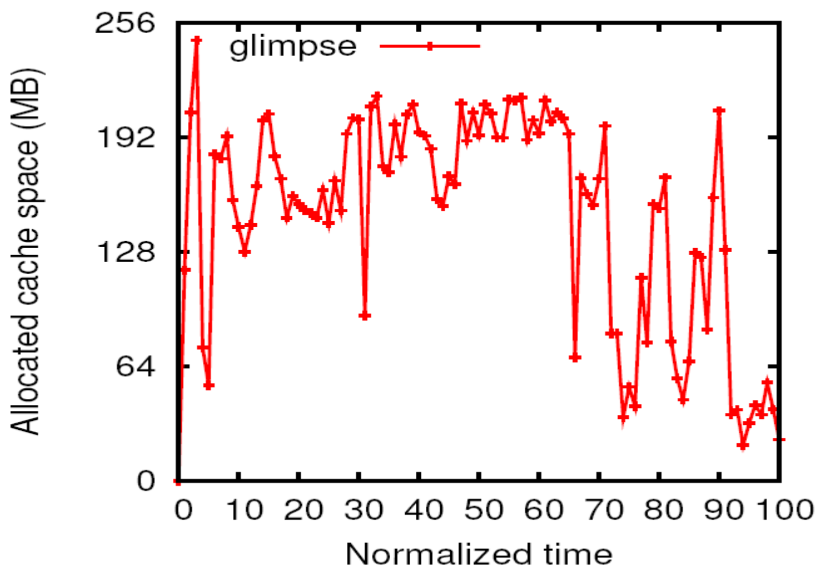
- How to provide performance guarantees and satisfy service level objectives (SLOs) of the applications ?
  (SLOs input to Neville's Algorithm)

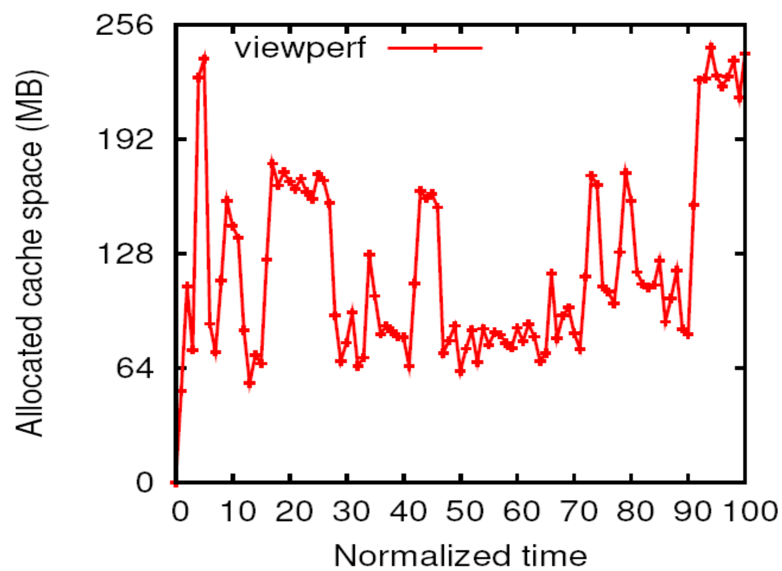
Overview of Storage Cache Partitioning Strategy

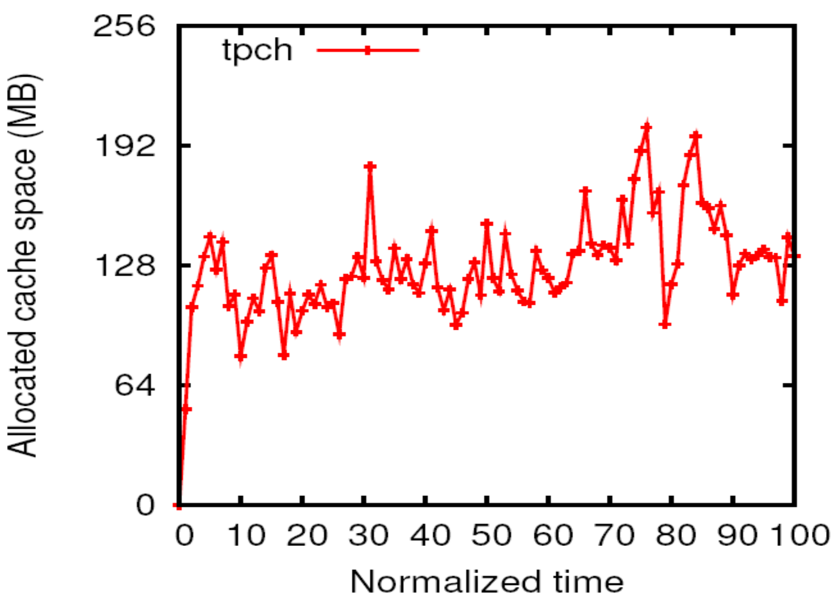


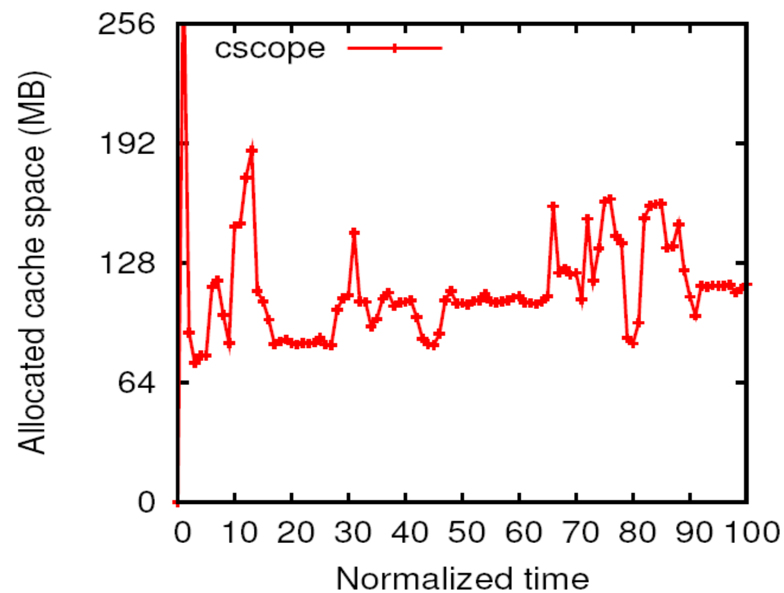
[Supercomputing 2010]

Cache Space Allocations over Time

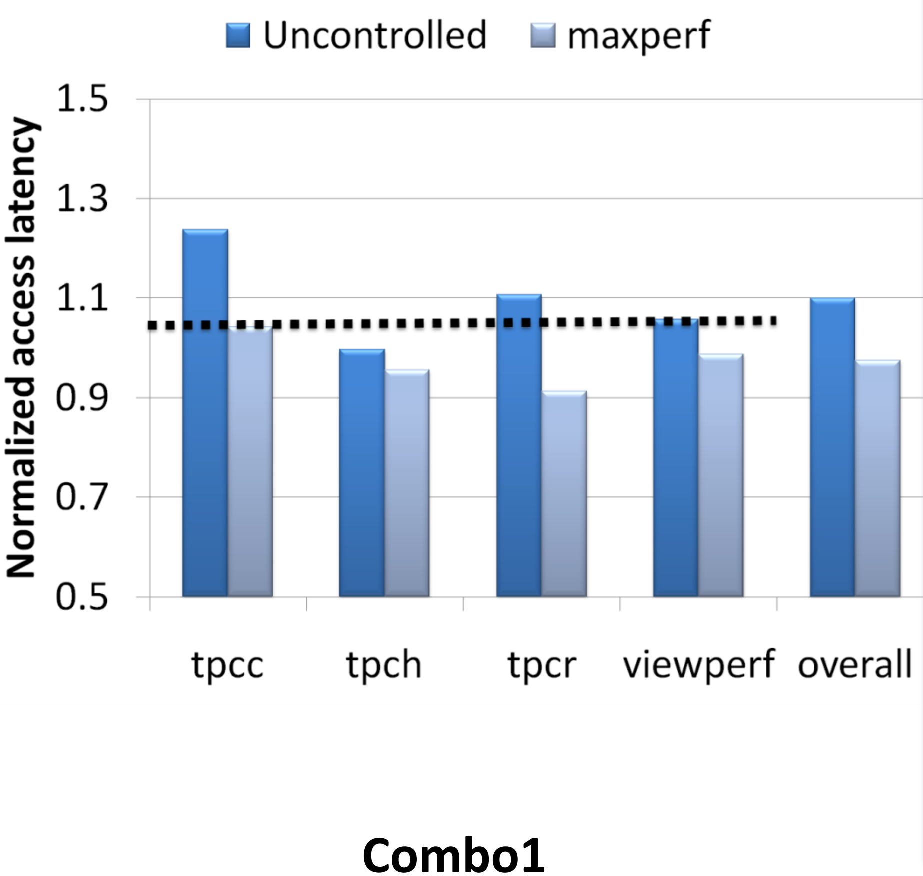


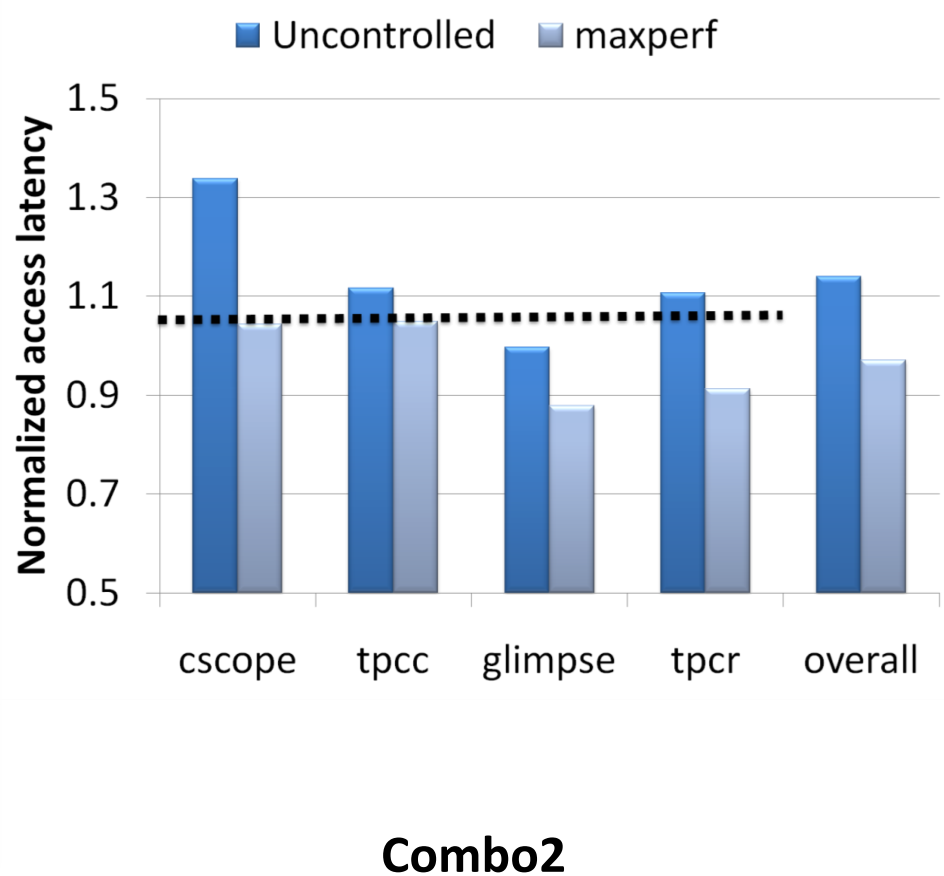






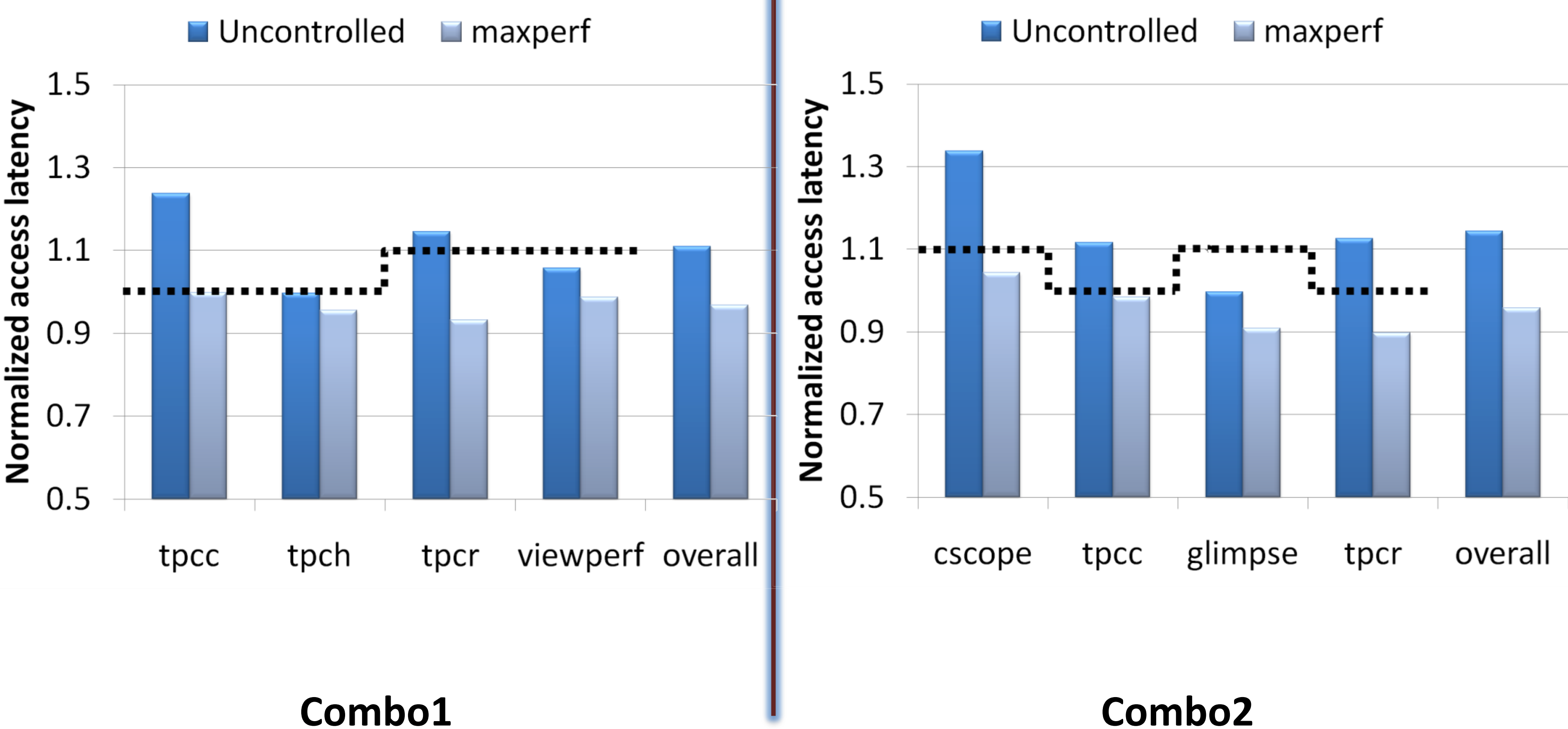
Results and Comparison





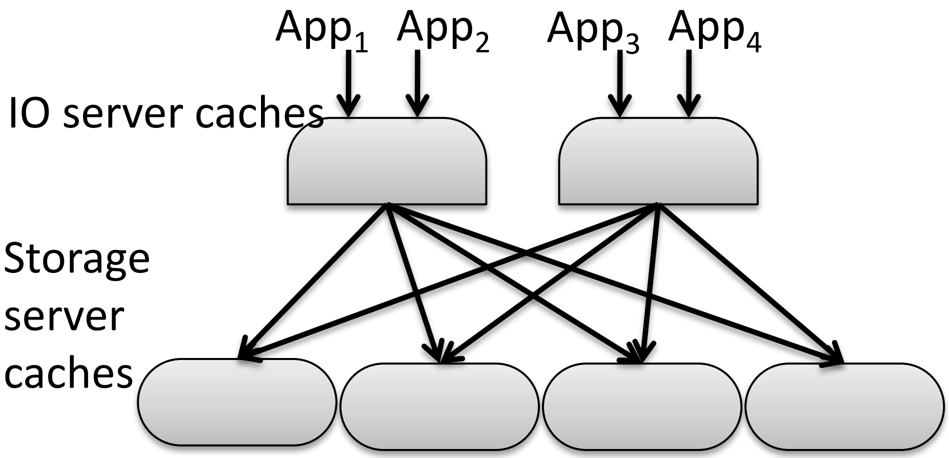
Overall latency improves by 19.6%

Change in SLO Specification

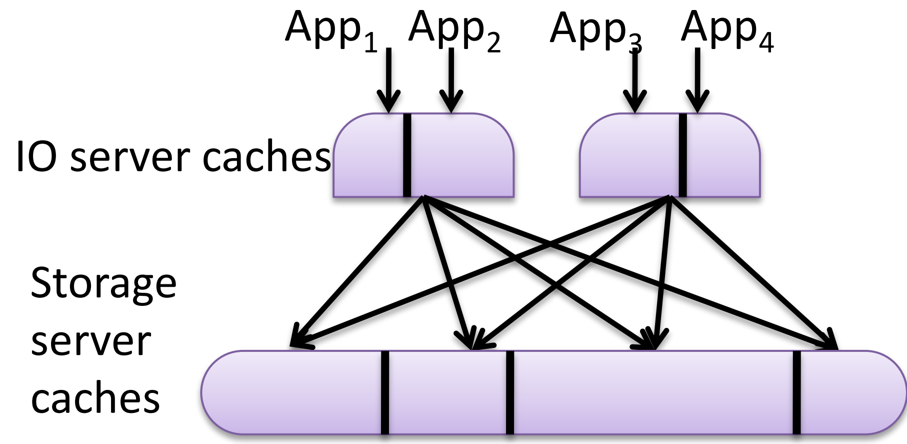


The overall I/O latency improves by up to 20.8%

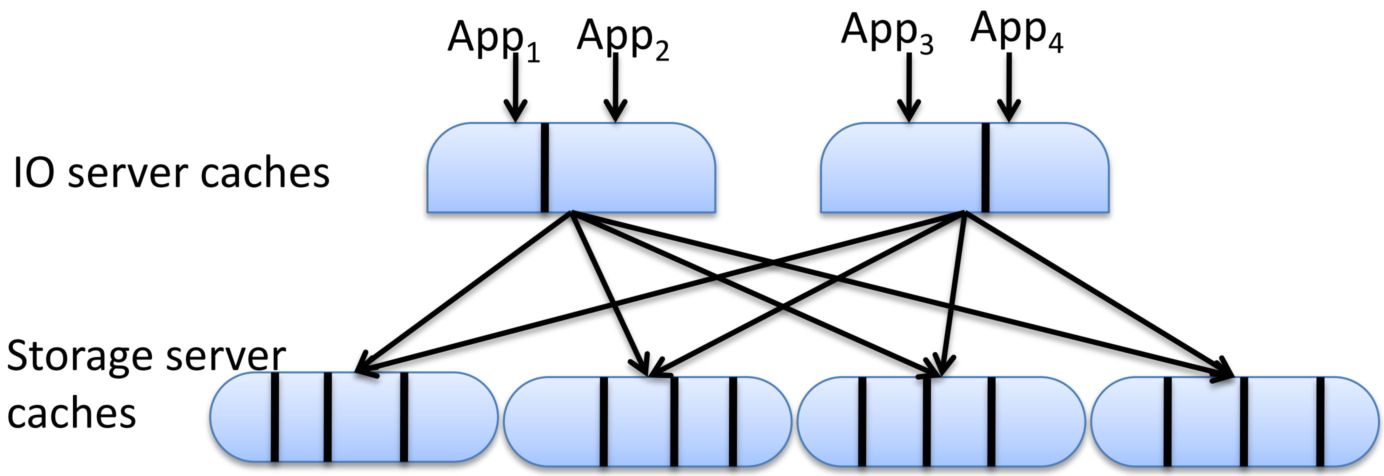
Multi-Level Multi-Server (MLMS) Algorithm



a) Default System



b) Multi-level Partitioning

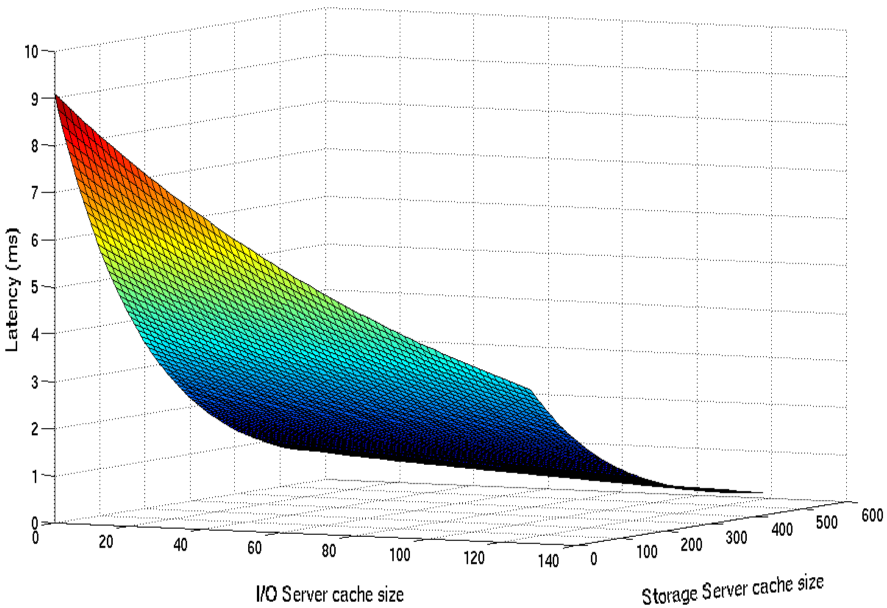


c) Multi-server Partitioning

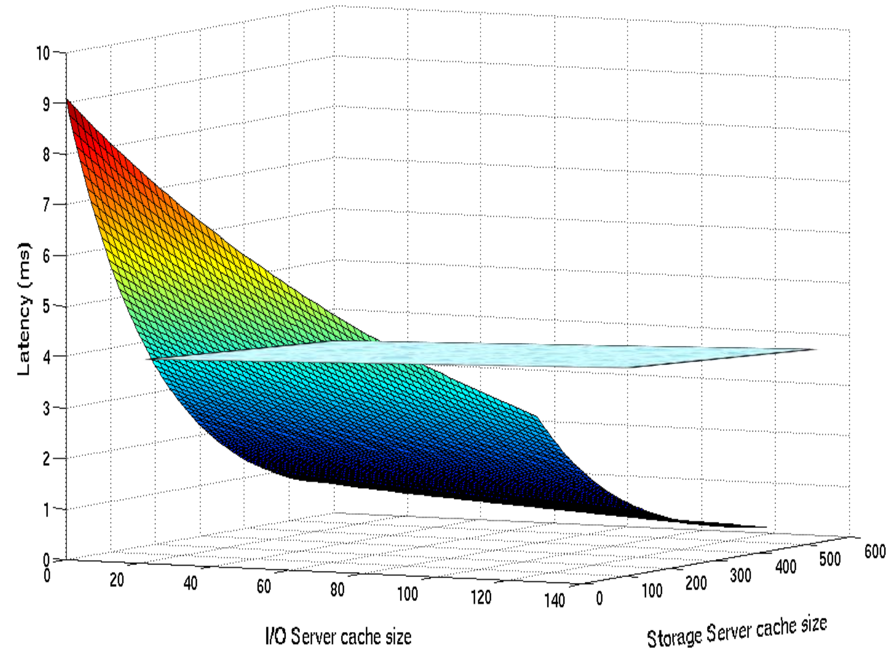
[ICS 2011]

Multi-Level Partitioning

i) Application Characterization



ii) SLO Fulfillment

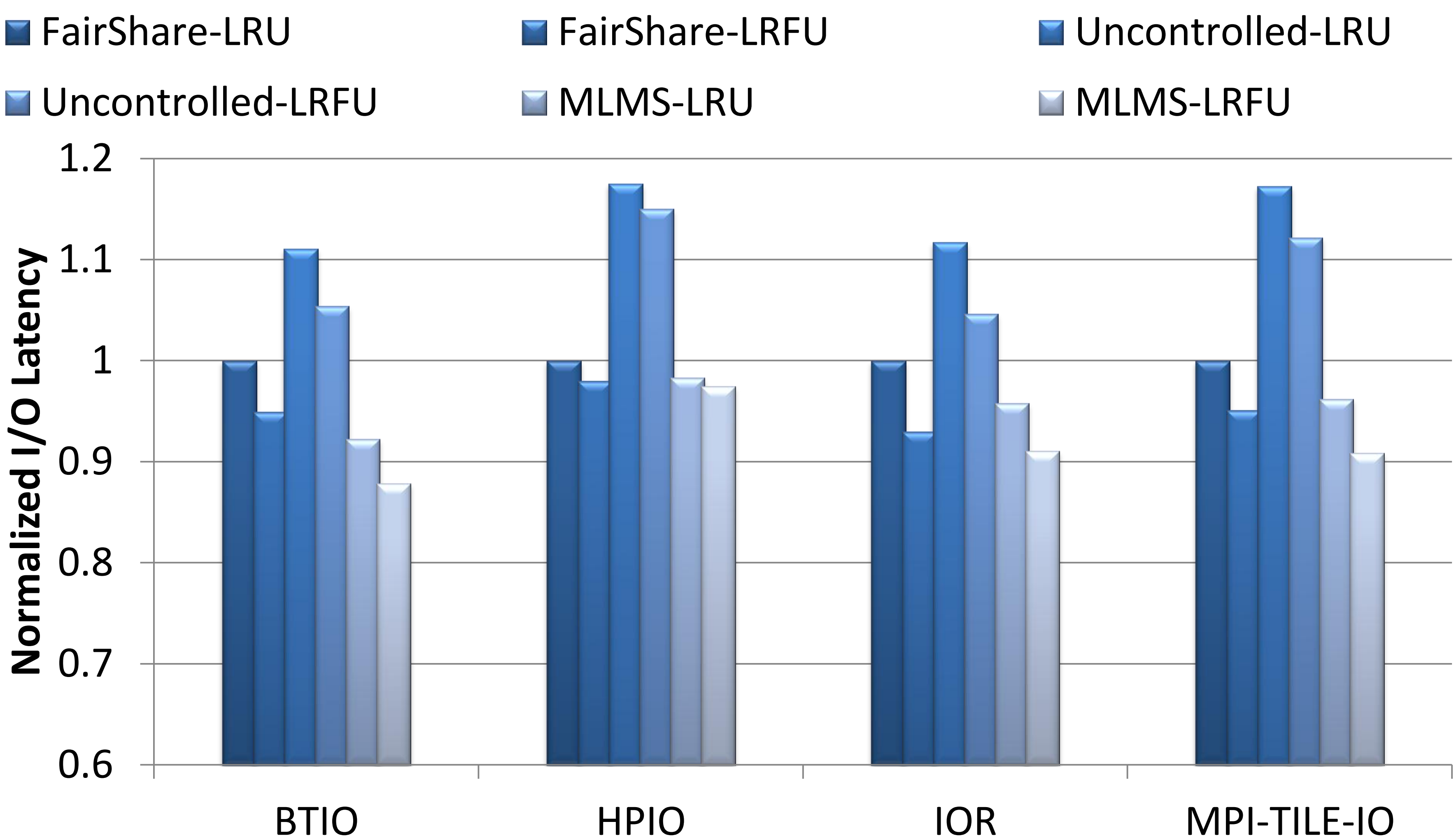


iii) **Determine the feasibility set**: Points that satisfy the SLO of all the applications and are within the physical cache constraints.

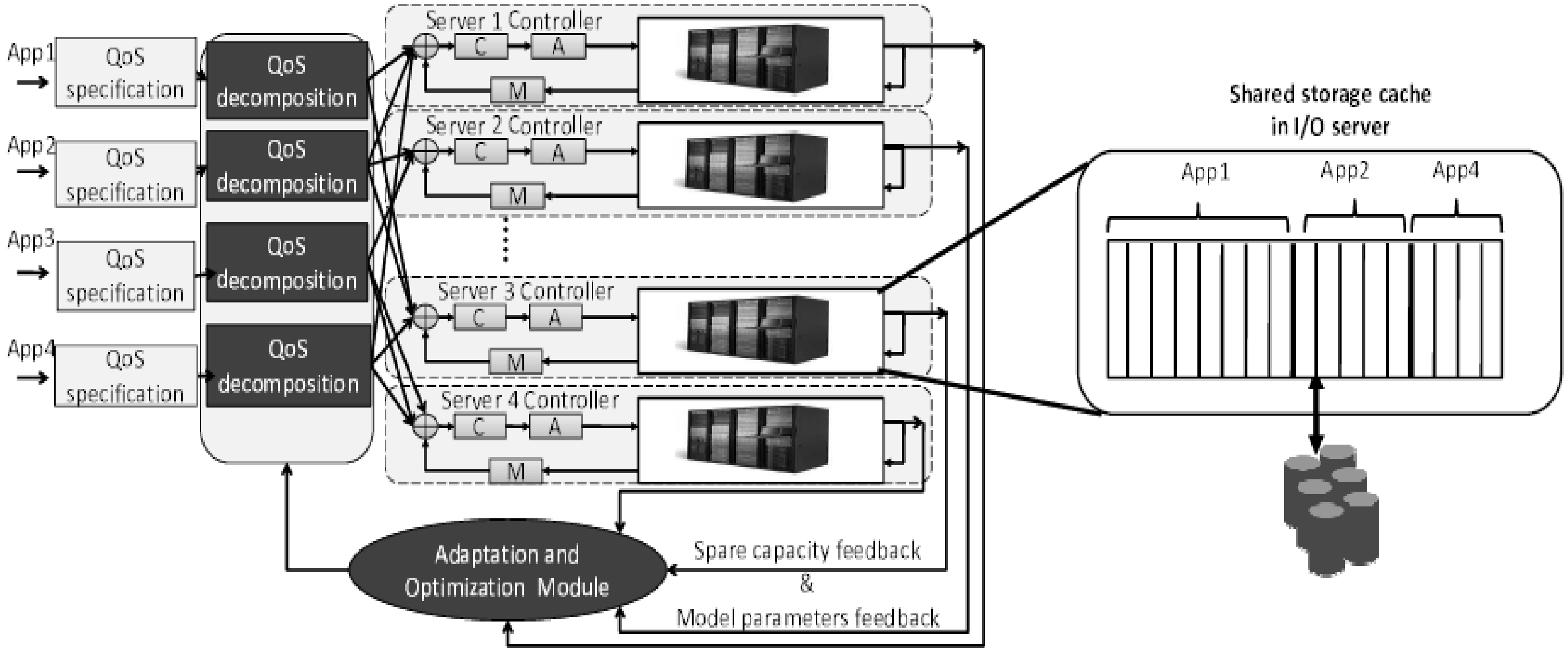
iv) **Maximize the fair-speedup metric (FS)**: Harmonic mean of per application I/O latency improvement with respect to the base scheme (fair share):

$$FS(scheme) = N \,/\, \sum_{j=1}^{N} IOAppj(base) \,/\, IOAppj(scheme)$$

Experimental Results



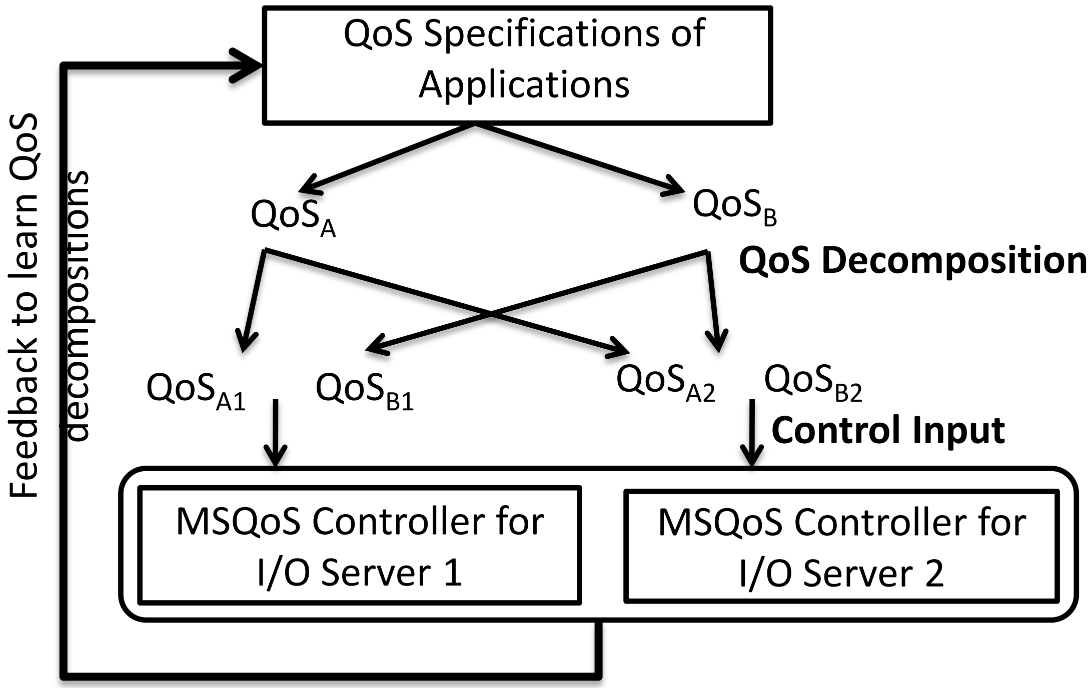
QoS Decomposition and Feedback Control



- ❖ Determines the best QoS decompositions using feedback from each I/O server.
- ❖ The components C, A and M correspond to Controller, Actuator, and QoS Monitor.
- ❖ Each I/O server manages allocation of storage cache among applications accessing it.

[CCGRID 2011]

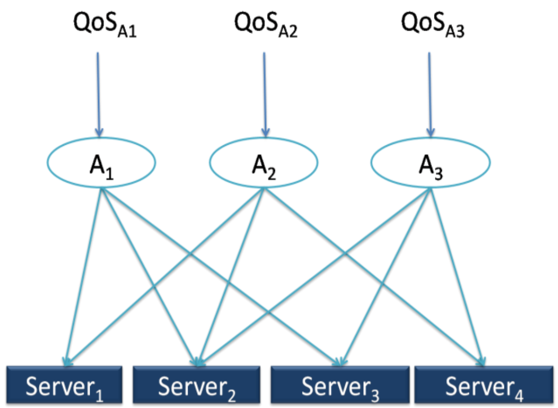
Interactions on Server Nodes



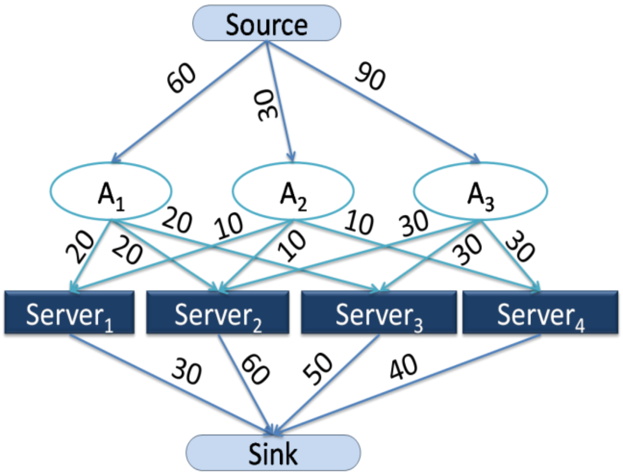
- ❖ Two levels of adaptations in our scheme in a two server system.
- ❖ QoS_{Xi} refers to the sub-QoS of application *X* that has to be satisfied from Server *i*.
- ❖ MSQoS controller manages resources, while decomposition module provides feedback on best QoS decompositions using the max-flow algorithm.

Adaptive QoS Decomposition

QoS specification

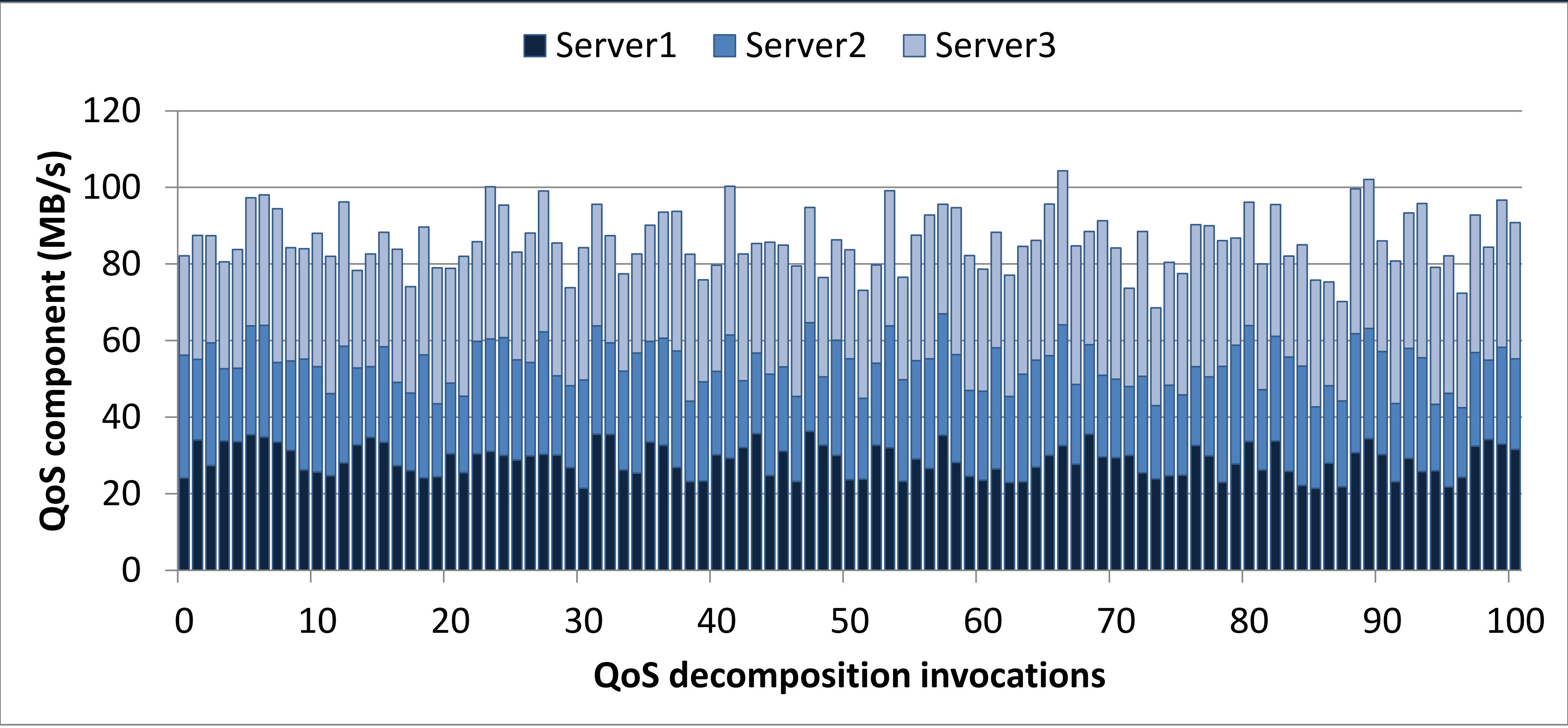


Max-flow algorithm



- ❖ Model the QoS decomposition as a network by adding virtual source and sink
- ❖ Applications and I/O servers form the vertices in the network
- ❖ Run max-flow algorithm

Illustrating QoS Decomposition



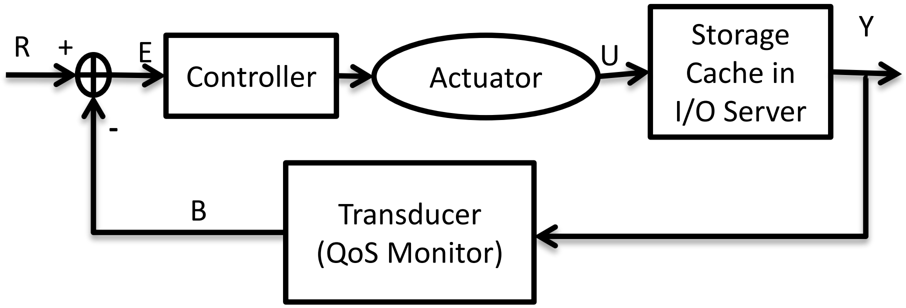
- QoS decomposition of MPI-Tile-IO into servers 1, 2, and 3.

- The dynamics of the decomposition are shown over 100 invocations of the adaptive QoS decomposition scheme.

Feedback Control Theory

Proposed MSQoS controller has **three** main components:

1) Each feedback control loop forms the **Per-Application Controller (PAC) component** of the controller.

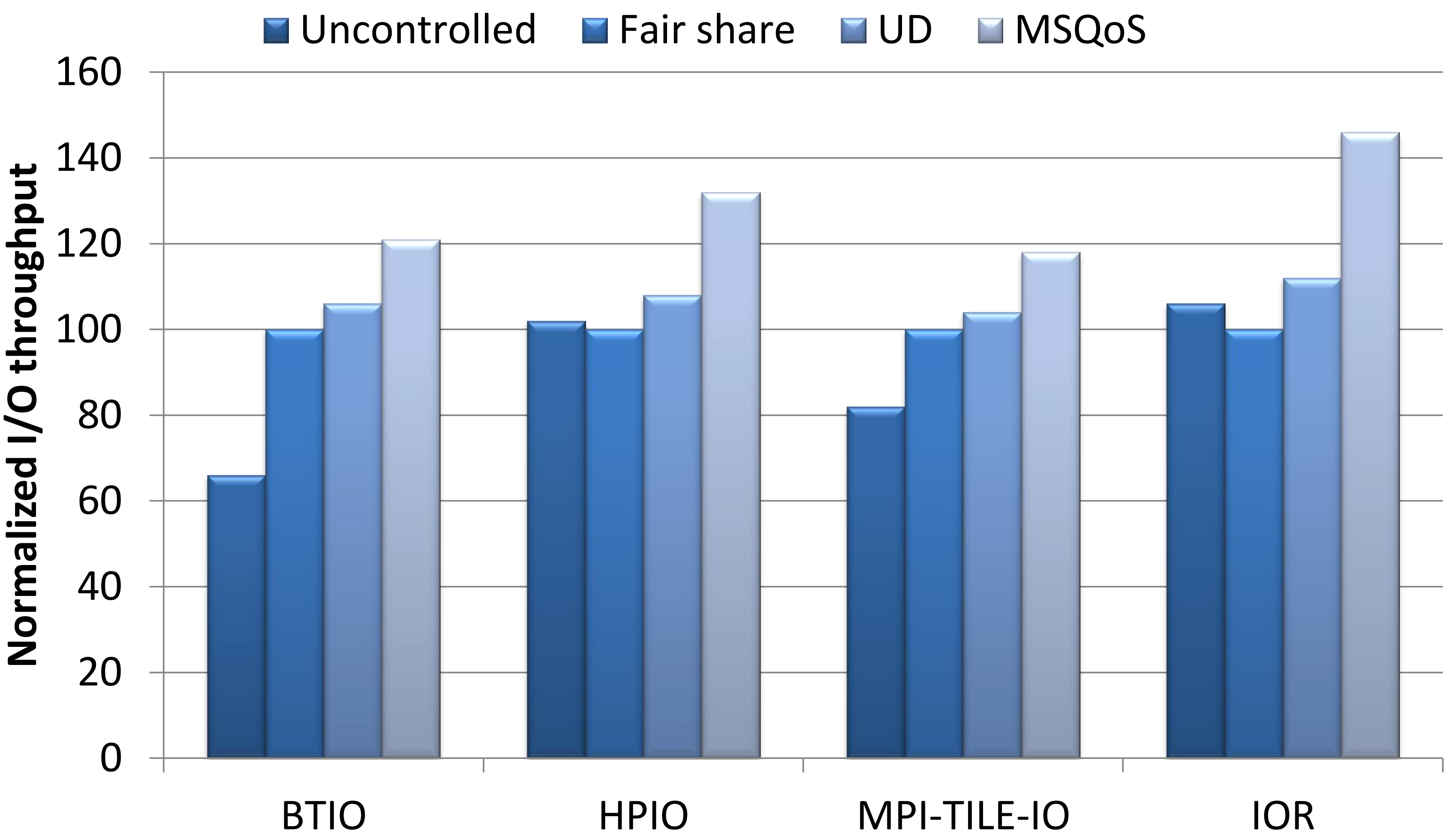


A simple feedback **control loop** for multi-server storage cache management problem.

2) Conflict manager (CM) provides a feasible allocation in each I/O server

3) Target revision component (TRC) of the controller increases the utilization of the storage cache

Experimental Results



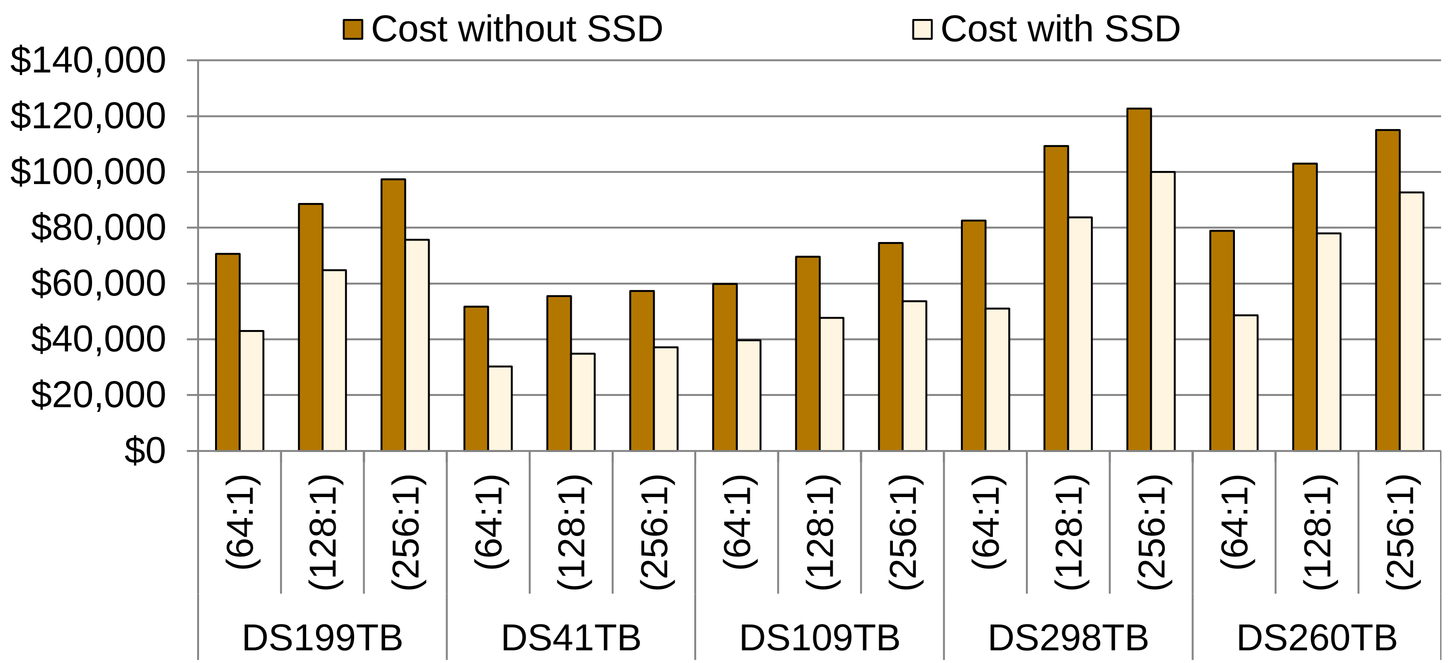
MSQoS improves the throughput of our applications by 48.6%, 29.2%, and 20.7%, respectively, over the uncontrolled partitioning, fair share and uniform decomposition schemes.

SSD Based Provisioning for Checkpointing

- **Staging area**
  - How an intermediate staging area can be positioned in the HPC center to absorb intense checkpoint data
- **Hybrid architecture from node-local resources**
  - A novel multi-tiered staging storage using node-local DRAM and SSD
- **Provisioning and cost/performance model**
  - A provisioning scheme to choose the least-cost storage configuration to meet performance goals
- **Evaluation**
  - Using a large-scale (2400-core) test-bed
  - Simulation study based on six years worth of Jaguar job logs

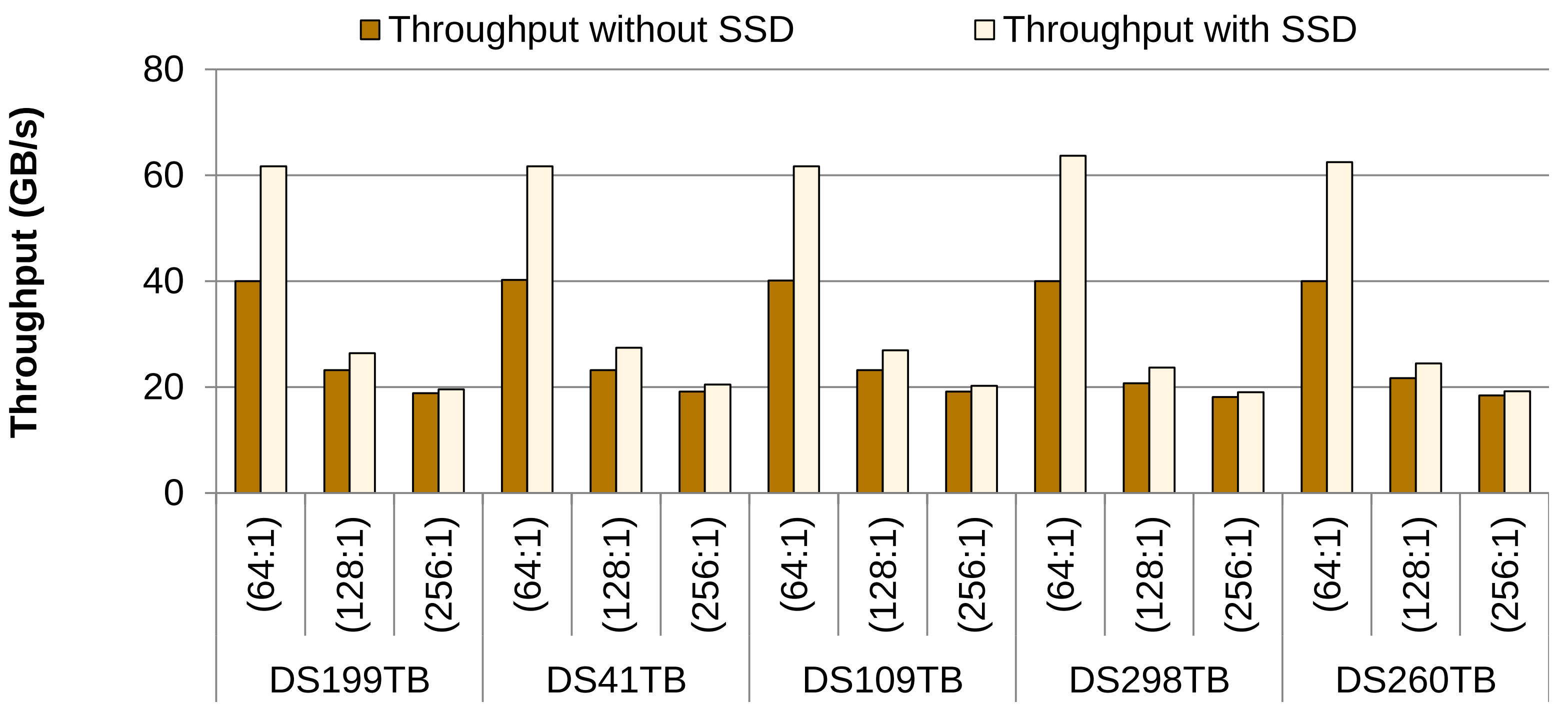
Ramya Prabhakar, Sudharshan Vazhkudai , et al [ICDCS 2011]

Cost Savings in the Staging Area



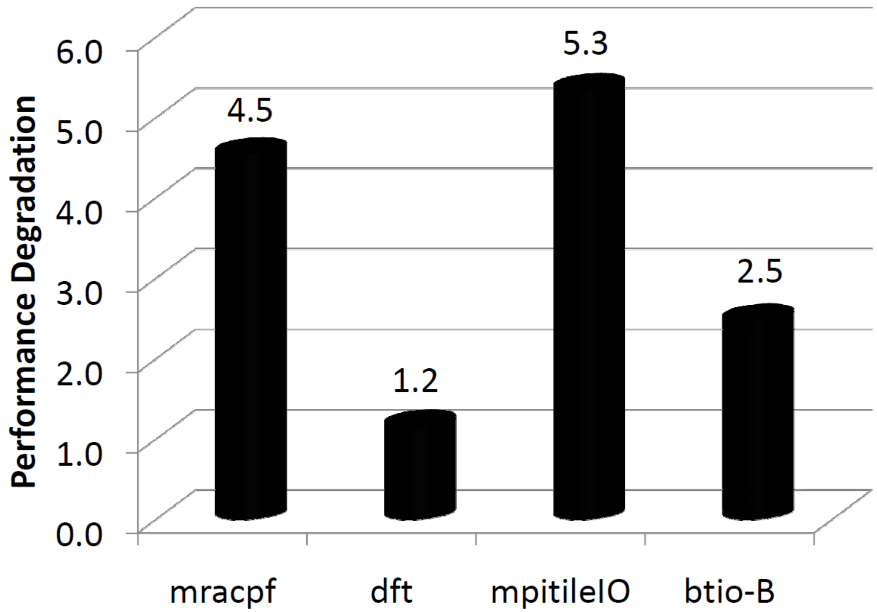
- Five “Hero” jobs, using 200,000+ cores:
  - Center-wide staging model
  - Compute node to staging Node ratios:
    - 64:1, 128:1, 256:1 similar to compute:I/O node ratios
  - If the staging storage is to sustain a throughput of checkpointing in 5 minutes every hour
    - 41.5% cost savings due to our provisioning scheme

Performance under Budget Constraints

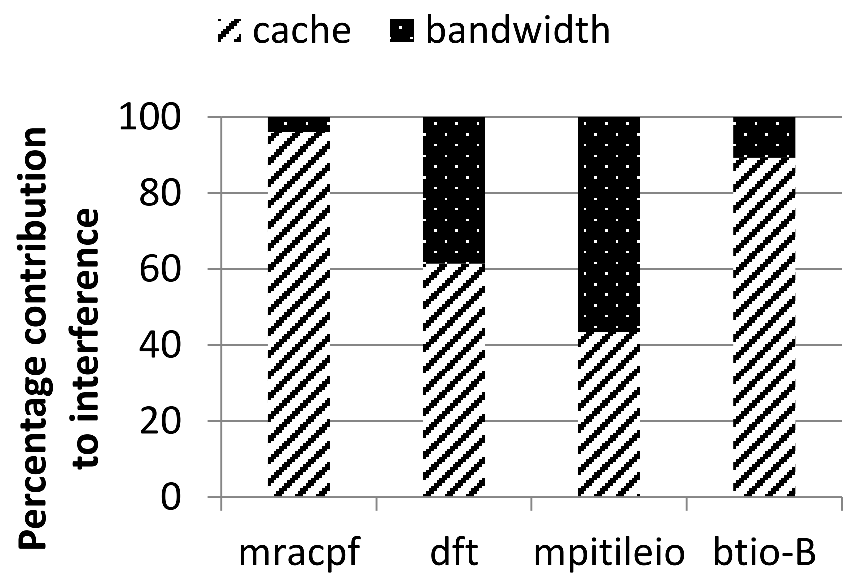


- Budget constraints:
  - For a budget of $90K
    - 59% improvement in throughput in using SSDs in the staging area

Ongoing Work: Coordinated Multi-Resource Partitioning



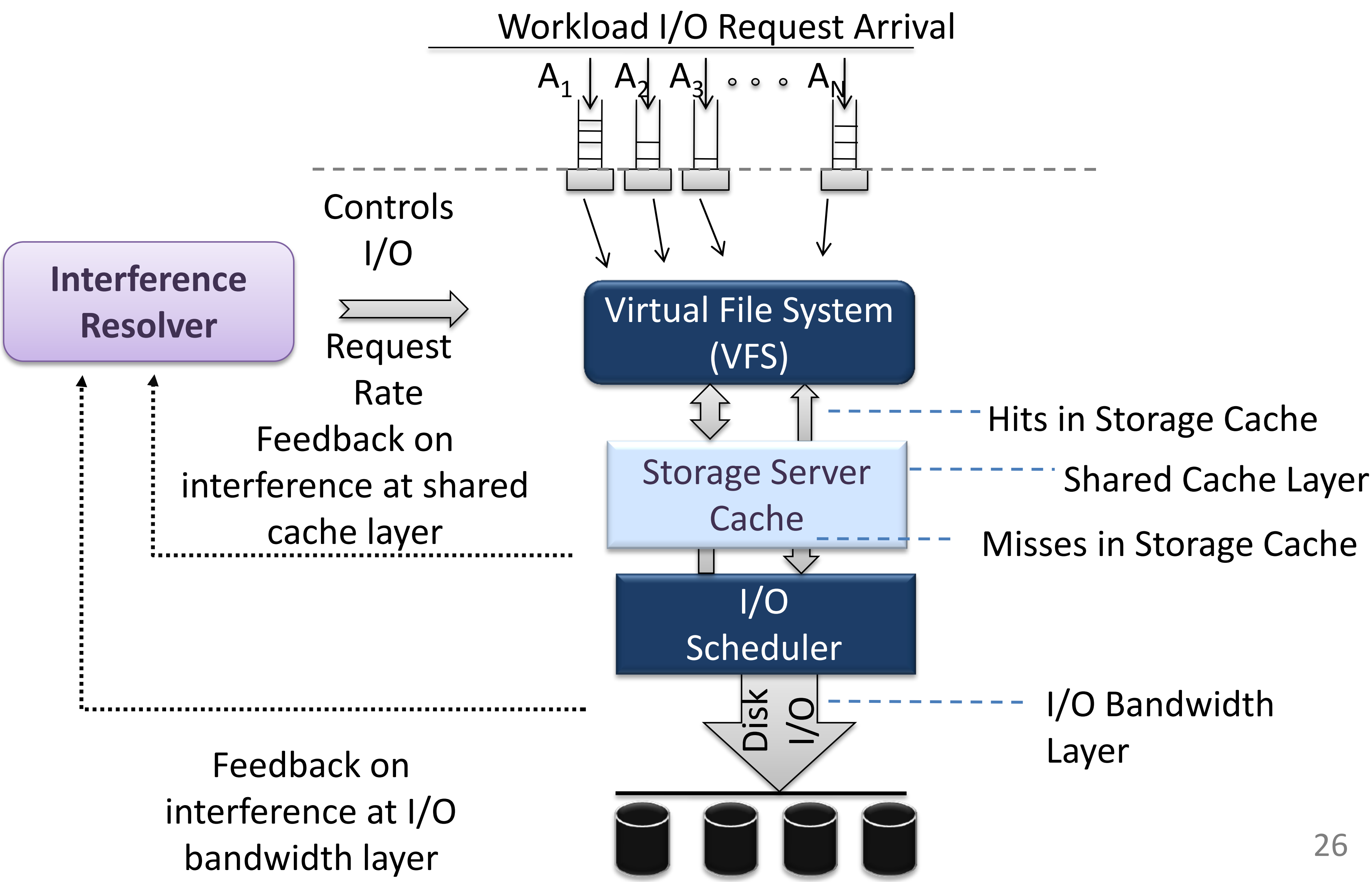
Performance degradation of different concurrently-executing applications on a shared storage system.



Quantifying interferences at both storage cache and I/O bandwidth levels

- Allocation decisions at storage cache directly influence the demand placed on I/O bandwidth
- Independently managing each resource may lead to contradictory decisions
- Motivates need for coordinated resource management scheme

Interference Resolver



Future Work

- End-to-End QoS in hybrid systems
  - Storage subsystem wide
  - Entire architecture wide
- Implementing different feedback control based strategies
- Testing with larger configurations